RECEIVED
CENTRAL FAX CENTER

JUN 05 2007

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named Applicant: Gutsche	Art Unit: 2166
Serial No.: 10/756,123	Examiner: Channavajjala
Filed: January 12, 2004	HSJ920030256US1
For: GUI FOR DATA PIPELINE	June 5, 2007 750 B STREET, Suite 3120 San Diego, CA 92101

APPEAL BRIEF

Commissioner of Patents and Trademarks

Dear Sir:

This brief is submitted under 35 U.S.C. §134 and is in accordance with 37 C.F.R. Parts 1, 5, 10, 11, and 41, effective September 13, 2004 and published at 69 Fed. Reg. 155 (August 2004). This brief is further to Appellant's Notice of Appeal filed herewith.

Table of Contents

06/07/2007 NNGUYEN1 00000002 502587 10756123
02 FC:1402 500.00 DA

(1) Real Party in Interest

The real party in interest is Hitachi Global Storage Technologies, Netherlands, B.V.

(2) Related Appeals/Interferences

No other appeals or interferences exist which relate to the present application or appeal.

(3) Status of Claims

Claims 1-9 are pending and finally rejected, which rejections are appealed, and claims 10-25 have been canceled.

(4) Status of Amendments

No amendments are outstanding.

(5) Summary of Claimed Subject Matter

As an initial matter, it is noted that according to the Patent Office, the concise explanations under this section are for Board convenience, and do not supersede what the claims actually state, 69 Fed. Reg. 155 (August 2004), see page 49976. Accordingly, nothing in this Section should be construed as an estoppel that limits the actual claim language.

The sole independent claim at issue (Claim 1) recites a graphical user interface (GUI) (reference numeral 310, figure 11; page 16, line 21) for configuring data pipelines (reference numeral 10, figure 1; page 6, line 7). Claim 1 requires that the GUI be displayable on a user computer monitor and stored on a computer

memory (page 7, lines 10 and 11). The GUI includes a pipe input set window (e.g., as shown in figures 18 and 19 and discussed on pages 22 and 23) configured to permit a user to define a type of pipe input set data, and a GUI page based on the type. The GUI page is generated by translating the type using a configuration file to a class and using Java reflection to generate an instance of the class (last line of page 22 through first three lines of page 23), and the instance produces the GUI page (last line of page 22 through first three lines of page 23). The GUI page is then used to configure a data pipeline (page 16, lines 9 and 10).

(6) **Grounds of Rejection to be Reviewed on Appeal**

(a) Claim 1 has been rejected under 35 U.S.C. §101 as being non-statutory.

(b) Claims 1-9 have been rejected under 35 U.S.C. §103 as being unpatentable over Blaszczak et al., USPP 2004/0186915 (hereinafter "ref a") in view of Yamamoto et al., USPN 6,311,151 (hereinafter "ref b").

(7) **Argument**

As an initial matter, it is noted that according to the Patent Office, a new ground of rejection in an examiner's answer should be "rare", and should be levied only in response to such things as newly presented arguments by Appellant or to address a claim that the examiner previously failed to address, 69 Fed. Reg. 155 (August 2004), see, e.g., pages 49963 and 49980. Furthermore, a new ground of rejection must be approved by the Technology Center Director or designee and in any case must come accompanied with the initials of the conferees of the appeal conference, id., page 49979.

a. Section 101 Rejection

Claim 1 has been rejected under 35 U.S.C. §101 as being non-statutory. Specifically, the examiner alleges that Claim 1, despite producing a GUI that is displayable on a computer monitor and used to configure a data pipeline, which the first paragraph of the background makes clear is useful for, e.g., continuously processing data in large scale manufacturing processes, grocery store sales, and the like, nonetheless lacks a "practical application." But the rejection is merely conclusory. It utterly fails to explain why a GUI, much less one that is explicitly disclosed to be displayed on a computer monitor for configuring a useful, tangible, and concrete data pipeline, lacks a practical application when it so obviously is highly practical.

Underpinning all of the rejections at bottom appear to be (although the grammar of the rejections makes it difficult to be certain) bare unsupported allegations that graphical user interfaces are mere software and hence non-statutory since they do not represent useful, tangible, and concrete results. This is a somewhat interesting position for the Patent Office to take. A GUI is not mere music, literary work, or simple collection of data as contemplated by MPEP §2106.01. Anyone who operates a computer these days uses a GUI to do so. It is a tool without which almost no PC would be operable.

Moreover, the Patent Office has issued over 700 patents with "GUI" or "graphical user interface" in the title. Plainly, producing a tool such as a hammer must be considered to produce a useful, tangible, and concrete result, and hammers are less ubiquitously used than GUIs. Not surprisingly, the allegation that producing a GUI is non-statutory is absolutely without a shred of legal authority: it is simply an unfounded and completely unexplained position taken by an examiner.

Nevertheless, in the interest in promoting prosecution Claim 1 was amended to positively recite in its body what had been suggested previously in the preamble, namely, that the GUI page is used to configure a data pipeline. Configuring a useful data pipeline, a few example useful applications of which are discussed in the first paragraph of the present background, cannot legitimately be said not to be a useful and tangible result.

This has been responded to by an odd *non-sequitur*, namely, an irrelevant comment (incorrect to boot, as it turns out) that a computer medium was not taught in the specification. This of course is a Section 112 argument, not a Section 101 argument, which in any case utterly fails to rebut any of the points made above. The response of the examiner to the above points concludes with a resolute refusal to face Appellant's points, merely restating the incorrect allegation that no "real-world" output that is useful is produced, Office Action, page 12, lines 7-10, despite the fact that a data pipeline is produced and its "real-world" uses divulged in the specification as pointed out above.

b(1) Obviousness Rejections - All Claims

Claims 1-9 have been rejected under 35 U.S.C. §103 as being unpatentable over ref (a), alleging that paragraphs 66 line 12 through paragraph 67 line 11 teach a pipe input set window configured to permit a user to define a type of pipe input set data and that paragraph 78 teaches producing a GUI by translating the type using a configuration file to a class, in view of ref (b), alleging that col. 5, lines 56-60 teach Java reflection.

Turning first to ref (a), the relied-upon portions state nothing at all about defining an input set data type as claimed. Instead, in pertinent part the relied-upon portion of paragraph 66 teaches simply that a node in the graph represents a specific predefined data transformation functionality that is offered by uninstantiated

component objects 370 residing in a component library 316 - not that a *type* of a data set, as opposed to a transformation, is defined, and less still that the GUI page is generated by translating the type using a configuration file to a class as claimed.

This has been responded by a representation that "examiner interpreting (sic) pipeline[s] corresponds (sic) to Blaszczak's data transformation pipline (sic) or DTP as detailed in fig 6A, element 302, and graphical user interface or GUI corresponds to Blaszczak's GUI fig 6A, element 304", Office Action, page 13, lines 3-6. As best understood by Appellant, this is nothing more than a statement that the examiner thinks Blaszczak et al. has a data pipeline and a GUI. Appellant's specific technical analysis of the shortcomings of the reference remain unrebutted.

Appellant notes that the examiner has not provided a claim construction for "data type" to aid Appellant in understanding the basis of the rejection (e.g., is it the examiner's position that "data type" encompasses compiler commands?), but in any case Appellant notes that claim terms must be construed as one skilled in the art would construe them, MPEP §2111.01, and no evidence has been pointed to in the references that skilled artisans regard compiler commands or file names to be "data types".

The relied-upon portion of paragraph 67 is of no further avail, because it too fails to state anything about defining a type for an input data set. Instead, it merely teaches that after graph data is input via the GUI, a translator translates the graph into an DFE plan. The translator also works in conjunction with an optimizer subsystem to optimize the graph into a maximally efficient execution structure by eliminating redundancies, simplifying and enhancing the DFE plan and possibly performing other optimizations - but not by defining a type of an input data set. For this reason, the rejections merit reversal.

Not surprisingly, the examiner disagrees with this too, Office Action, page 13, line 10. However, as was the case above, the examiner fails to take issue with the specific technical analysis made by Appellant regarding paragraph 67, instead making additional generalizations to the effect that the reference teaches modules and APIs, and that it uses a particular type of tool - ***not***, however, that it teaches anything at all about defining a type for an input data set as claimed.

In addition, Appellant would like to point out that paragraph 78 of ref (a) does not, contrary to the allegation in the rejection, produce a GUI by translating the type using a configuration file to a class. Nothing in paragraph 78 implicates a type of an input data set, much less to use the type to produce a GUI, much less still in the particular way claimed. Instead, paragraph 78 further discusses the importance of the translator/optimizer portions discussed above (which are parts of a "scheduler"), and that the scheduler further controls actual execution of data flows. For this further reason, the rejections merit reversal.

As usual, the examiner disagrees without specifically supporting a contrary position beyond a broad-brush allegation, in this case, that the reference generally teaches "consistent functionality". If this in some mysterious way actually addresses the very specific and technical issue being discussed (translating the type using a configuration file to a class), Appellant has been unable to divine it.

Still further, the suggestion to modify ref (a) with the alleged "Java reflection" of ref (b) to arrive at Claim 1 is misplaced. Specifically, the relied-upon portion of ref (b) simply states that each Java bean component has its own features, including its properties, methods, and events, and that the features can be examined by other beans by a process known as introspection, which is supported in two ways. First, ref (b) teaches that specific rules, known as design patterns, are followed when naming bean features. Ref (b) explains that a particular class examines beans for these design patterns to discover bean features, and that this

class relies on the "core reflection API." But this API is never further mentioned, either in terms of what it is or what it does.

Accordingly, to the extent that the "core reflection API" is the same thing as "Java reflection", which Appellant does not concede, it is used at most to examine the features of Java beans. In contrast, Claim 1 requires using Java reflection for something entirely different and, hence, unsuggested by ref (b), namely, generating an instance of a class of a type of pipe input set data. Thus, ref (b) does not suggest anything remotely close to a modification of ref (a) that would reach Claim 1, further rendering the rejection overcome.

The Board will not be surprised to learn that the examiner disagrees. It will also not come as a surprise that the disagreement is expressed in generalized terms without addressing the specific technical issue at hand. More particularly, on page 14 of the Office Action, part (d), it is repeated that Blaszczak et al. teaches a data pipeline that transforms data, and that its pipeline can use Java. On page 15 it is alleged that Yamamoto et al. teaches using Java reflection to generate "an instance of a class", relying on col. 5, lines 55-60, but this simply wrong. The relied-upon portion of ref (b) teaches only that a "core reflection API" is used to "examine Beans for design patterns to discover Bean features". The Office Action then descends into what a "reflection API" *typically* does and what it allegedly *represents* without any prior art support for the contentions and, more importantly, without ever getting round to showing why examining "Beans" for features as is done in ref (b) is the same thing as using Java reflection to generate an instance of the class which produces a GUI page as claimed.

b(2) Obviousness Rejections - Dependent Claim 2

With respect to Claim 2, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraph 82 teaches a pipe input set window and GUI page that require no programming apart

from an initial core code. All this part of ref (a) teaches is the details of how the translator/optimizer translate the graph into a DFE.

This has been responded to with a nearly incomprehensible statement, the import of which appears to be that paragraph 82 teaches building from a library and thus for some reason meets the limitations at issue, which of course nowhere mention "library" but which do recite some very detailed technical features that remain studiously ignored.

b(3) Obviousness Rejections - Dependent Claim 3

With respect to Claim 3, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraph 82 teaches that the GUI is an incremental GUI wherein GUI pages for new pipe components can be added incrementally without changing existing code. All this part of ref (a) teaches is the details of how the translator/optimizer translate the graph into a DFE.

This observation and those following for Claims 4 and 6 have been castigated for being "merely conclusory" for failing to address "examiner's particular interpretation of the reference". That's the problem - no claim interpretation has ever been provided by the examiner nor has he explained why simply declaring that the apple of the reference is the seemingly unrelated orange of the claim has merit. Appellant is not playing word games "interpreting" what the reference teaches in a way that conveniently suits its case, but instead is focusing on what the reference ***actually teaches***, which is something very different than what the claim recites.

b(4) Obviousness Rejections - Dependent Claim 4

With respect to Claim 4, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraph 9 teaches that a new pipe module is based on a pre-existing module type. All this portion of ref (a) teaches is that GUI graph can be used to develop a DFE, with each node in the graph defining a transformation function.

b(5) Obviousness Rejections - Dependent Claim 6

With respect to Claim 6, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraphs 69-73 teach that the GUI defines a set of interfaces, with each interface including plural functions. The relied-upon paragraphs do not describe interfaces much less ones that includes plural functions, but rather simply set out the five operations used to control the DFE of ref (a).

This observation has not been rebutted.

With further respect to Claim 6, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraphs 53 and 81 teach that the GUI includes a GUI representation part and a storage part, with the GUI representation part defining how something is displayed and the storage part defining how GUI parameters are stored in an external storage. Instead, paragraph 53 discusses exposing a user interface to an SQL server, not that a GUI includes two parts, much less the ones claimed, much less still what the parts define. Paragraph 81 likewise adds nothing of import, describing only how the GUI is used to define how to extract data from a database.

This has been responded to in relevant part by referring to Blaszczak et al., elements 304 and 380 and paragraphs 53 and 81. But the element 304 is simply a GUI, while the element 380 is a buffer, paragraph 76,

whereas Claim 6 requires more than a mere GUI with a storage. As required by the claim, the GUI itself has a GUI representation part and a storage part, with the GUI representation part defining how something is displayed and the storage part defining how GUI parameters are stored in an external storage. The relied-upon portions of the reference do not appear to contemplate that part of the GUI defines how GUI parameters are stored in the relied-upon buffer.

b(6) Obviousness Rejections - Dependent Claim 7

With respect to Claim 7, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraph 80 teaches a tab for defining a set representative of a type of output data from the pipeline. Paragraph 80 merely states that data remains in a buffer during DTP operations. No tab is mentioned, much less in the context of a GUI, much less still for the purpose recited in Claim 7. The examiner insists otherwise without support and without specifically attempting to address where the precise limitation of defining a set representative of a type of output data is to be found in the reference.

b(7) Obviousness Rejections - Dependent Claim 8

With respect to Claim 8, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraph 78 teaches a tab for defining an arbitrary number of elements contained in a component of the pipeline, with individual input and output sets being definable for each element in the component. Instead, paragraph 78 further discusses the importance of the translator/optimizer portions discussed above (which are parts of a "scheduler"), and that the scheduler further controls actual execution of data flows. There is no teaching in paragraph 78 of a tab, arbitrary number of elements in a component, or individual

input and output sets that are defined for each element in the component. The propriety of the examiner insisting otherwise without further support will be left to the Board to decide.

b(8) Obviousness Rejections - Dependent Claim 9

With respect to Claim 9, Appellant would like to point out that the allegation appears to be incorrect that ref (a), paragraph 7 teaches a tab for defining an arbitrary number of modules of the pipeline, with a type being selected for each module using the tab and with the type defining part of the GUI. Instead, paragraph 7 contains no teachings of tabs, much less tabs that are part of a GUI, or of selecting modules, much less doing so using GUI tabs. Paragraph 7 simply discusses scheduling data flow execution using a graph. The propriety of the examiner insisting otherwise will be left to the Board to decide.

Respectfully submitted,

John L. Rogitz
Registration No. 33,549
Attorney of Record
750 B Street, Suite 3120
San Diego, CA 92101
Telephone: (619) 338-8075

JLR:jg

1189-25.AIT

APPENDIX A - APPEALED CLAIMS

1. A graphical user interface (GUI) for configuring pipelines, the GUI displayable on a user computer monitor and stored on a computer memory and comprising:

at least one pipe input set window configured to permit a user to define a type of pipe input set data;

at least one GUI page based on the type, the GUI page being generated by translating the type using a configuration file to a class and using Java reflection to generate an instance of the class, the instance producing the GUI page; and

using the GUI page to configure a data pipeline.

2. The GUI of Claim 1, wherein at least the pipe input set window and GUI page require no programming apart from an initial core code.

3. The GUI of Claim 1, wherein the GUI is an incremental GUI wherein GUI pages for new pipe components can be added incrementally without changing existing code.

4. The GUI of Claim 3, wherein at least one new pipe module is based on a pre-existing module type.

5. The GUI of Claim 3, wherein at least one new pipe module is based on a new user-defined component type.

6. The GUI of Claim 1, wherein the GUI defines a set of interfaces, each interface including plural functions, the GUI including a GUI representation part and a storage part, the GUI representation part defining how something is displayed and the storage part defining how GUI parameters are stored in an external storage.

7. The GUI of Claim 1, further comprising:

at least one *Pipe Output Set* tab for defining *PipeOutputSet* representative of a type of output data from the pipeline.

8. The GUI of Claim 1, further comprising:

at least one *Storage For TupleSets* tab for defining an arbitrary number of elements contained in a *StorageForTupleSets* component of the pipeline, individual input and output sets being definable for each element in the component.

9. The GUI of Claim 1, further comprising:

at least one *Pipe Modules* tab for defining an arbitrary number of *PipeModules* of the pipeline, a type being selected for each *PipeModule* using the tab, the type defining at least in part the GUI.

APPENDIX B - EVIDENCE

None (this sheet made necessary by 69 Fed. Reg. 155 (August 2004), page 49978.)

APPENDIX C - RELATED PROCEEDINGS

None (this sheet made necessary by 69 Fed. Reg. 155 (August 2004), page 49978.)